Deploying to AWS Lambda from Github

Lambda overview

AWS Lambda is a serverless offering that allow you to run commands in many different languages based on events.

Deploying

After you have your lambda function setup you have several different options to deploy your code.

1. The simplelest way is to edit directly in the AWS console.
2. Upload zip of your code.

If you are like me you put all of your code in version control. In my case that is Github. Through the magic of Github Actions, you can deploy automatically triggered by pushing to the Main branch.

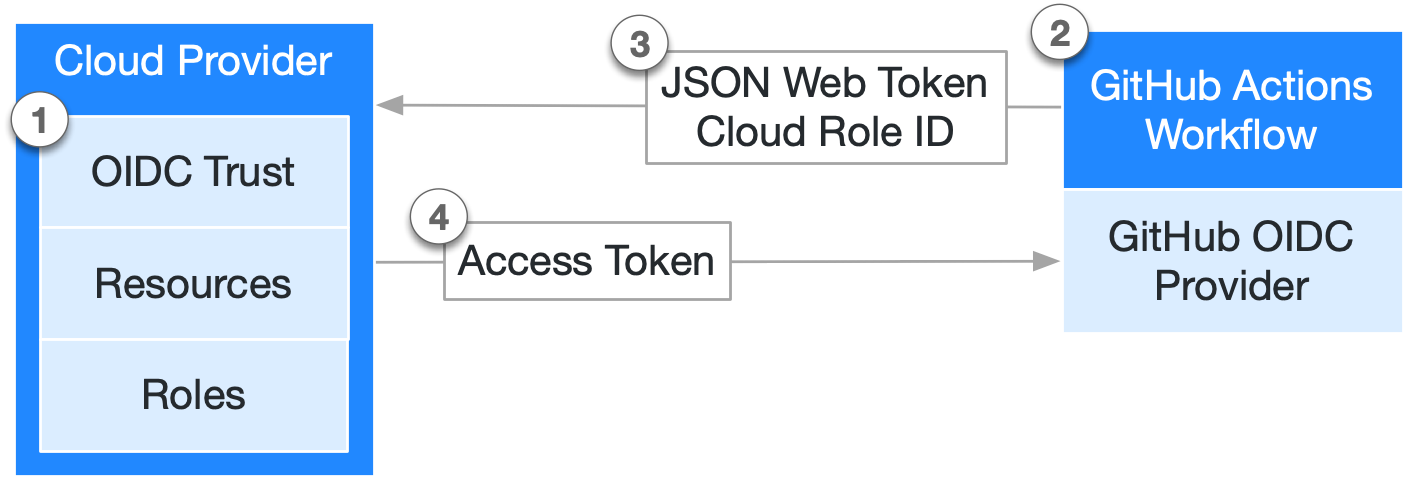
If you are not familiar, Github Actions is a way to automate your CI/CD piplines among other tasks. It consists of building blocks, Actions that can be combined create a workflow of jobs and steps. Each job actually runs on a linux, windows, or mac virtual machine. The workflow is configured by creating a yaml file. More information can be found here: <https://docs.github.com/en/actions>.

Github Actions ultimately uses AWS CLI to interact with your AWS account. This means that your Github Actions repo needs to authenticate with AWS. Just like anytime you use AWS CLI. You can use a AWS Key Id and Secret, generated by IAM, but this requires hardcoding them in Github Secrets. A better solution is to configure OpenID Connect (OIDC), a form of federation. Creating a trust between Github and your AWS account and having your Actions workflow request a short-lived access token with limited permission to the AWS resources.

Configuring OIDC has the following benefits

* Short-lived AWS tokens – no need to configure long term AWS Secrets. Instead OIDC uses tokens that by default last for one hour. This is configurable from 15 minutes up to 12 hours.
* Limited perissions – more granular control over AWS resources the token can access and what they can do to those resources.
* Rotating credentials – as a result of the short-lived tokens, the credentials expire and rotate after every job.

Let’s walk thru setting this up.



1. We begin by creating an OIDC trust between your AWS Account role and your GitHub workflow(s).
   * Within AWS IAM we need to create an OIDC Identity Provider. This will describe the GitHub external connection.
     1. Open Identity Providers in AWS IAM
     2. Click the Add Provider button and select OpenID Connect
     3. Enter your provider Url and click the Get Thumbprint (this is the cert thumbprint from the provider URL)
        1. In this case we will use <https://token.actions.githubusercontent.com>
     4. Enter the audience
        1. Sts.amazonaws.com
   * Now that we have the Identity Provider, we need to assign a role to it.
     + Click the Assign role button
     + You can then select an existing role or create a new one.
       - The role should have the permissions you want Github Actions to be able to perform so you need to assign a Permissions Policy to the role.
       - In this instance I allow Github Actions to Tag, Untag lambda as well as UpdateFunctionCode. But only on the resources I have specified by the Lambda Function ARNs listed.

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"lambda:TagResource",

"lambda:UntagResource",

"lambda:UpdateFunctionCode"

],

"Resource": [

"arn:aws:lambda:us-east-1:123456789000:function:SearchMediaGuide",

"arn:aws:lambda:us-east-1: 123456789000:function:WaterConditions",

"arn:aws:lambda:us-east-1: 123456789000:function:SendAWSSESMessage",

"arn:aws:lambda:us-east-1: 123456789000:function:json2table"

]

}

]

}

* In the same Role you need to set up the Trust Relationship tab
  + Edit the Trust policy to include the federated principal (this is the ARN for the Identity Provider you created earlier
  + What Action it can take
    - Sts:AssumeRoleWithWebIdentity
  + And a Condition
    - Audience = “sts.amazonaws.com”
    - And subscribers = any of your repos/branches you will allow through

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Federated": "arn:aws:iam::12345678900:oidc-provider/token.actions.githubusercontent.com"

},

"Action": "sts:AssumeRoleWithWebIdentity",

"Condition": {

"StringEquals": {

"token.actions.githubusercontent.com:aud": "sts.amazonaws.com"

},

"ForAnyValue:StringLike": {

"token.actions.githubusercontent.com:sub": [

"repo:GITHUBACCT/REPO1:ref:refs/heads/main",

"repo: GITHUBACCT/REPO2:refs/heads/main",

"repo: GITHUBACCT/REPO3:ref:refs/heads/main",

"repo: GITHUBACCT/REPO4:ref:refs/heads/main"

]

}

}

}

]

}

* + For additional information on setting up an Identity Provieder: <https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services>

1. Every time your job runs, GitHub's OIDC Provider auto-generates an OIDC token. This token contains multiple claims to establish a security-hardened and verifiable identity about the specific workflow that is trying to authenticate.
2. You could include a step or action in your job to request this token from GitHub's OIDC provider, and present it to the cloud provider.
3. Once the cloud provider successfully validates the claims presented in the token, it then provides a short-lived cloud access token that is available only for the duration of the job.

Now onto the fun part. Below is workflow I use to deploy my main branch to my AWS Lambda function

name: deploy to lambda

on:

  push:

    branches:

      - main

permissions:

  id-token: write

jobs:

  deploy\_source:

    name: build and deploy lambda

    strategy:

      matrix:

        node-version: [12.x]

    runs-on: ubuntu-latest

    steps:

      - name: Checkout Private Repo

        uses: actions/checkout@v3

        with:

          token: ${{secrets.ACTIONSTOKEN}}

      - name: zip

        uses: montudor/action-zip@v0.1.0

        with:

          args: zip -qq -r ./deploy.zip ./

      - name: Configure AWS Credentials

        uses: aws-actions/configure-aws-credentials@v1

        with:

          aws-region: us-east-1

          role-to-assume: arn:aws:iam::123456789000:role/GithubActions-Lambda

          role-session-name: GithubActions-JeffBuenting

      - name: AWS Lambda Deploy

        run: |

          aws lambda update-function-code --function-name WaterConditions --zip-file fileb://waterconditions.zip

Here is what each section means

|  |  |
| --- | --- |
| On: | means the workflow will run anytime a push completes on the main branch. |
| Deploy\_source: | This job runs on the latest version of Ubuntu container |
| Checkout Private Repo Step | Checks out the repo so we have access to the files |
| Zip Step | Zips the repos files |
| Configure AWS Credentials Step | This is the step that exchanges the JSON Github token with your accounts AWS Identity Provider configured above |
| AWS Lambda Deploy Step | AWS CLI command to deploy the zip to the lambda function |
|  |  |

So any time I push or complete a pull request to the main branch of the repo this action will run and the code will get securely deployed to my AWS Lambda Function.